Communication Efficient Gaussian Elimination with Partial Pivoting using a Shape Morphing Data Layout

Grey Ballard, James Demmel, Benjamin Lipshitz,
Oded Schwartz, and Sivan Toledo

SPAA
July 24, 2013

Funding from Microsoft (Award #024263) and Intel (Award #024894), and matching funding by U.C. Discovery (Award #DIG07-10227). Additional support comes from ParLab affiliates National Instruments, Nokia, NVIDIA, Oracle and Samsung, as well as MathWorks. Research is also supported by DOE grants DE-SC0004938, DE-SC0005136, DE-SC0003959, DE-SC0008700, and AC02-05CH11231, and DARPA grant HR0011-12-2-0016, and grant 1045/09 from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities), and grant 2010231 from the US-Israel Bi-National Science Foundation.

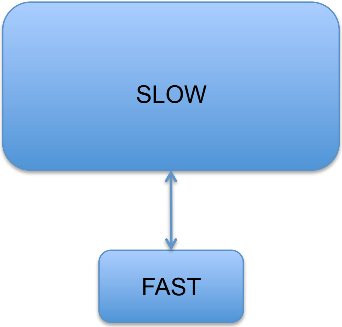
Summary

I'll present an algorithm for performing Gaussian elimination (i.e., computing an LU decomposition to solve a dense linear system) that

- is **communication optimal** and **cache oblivious**
  - matches the communication lower bounds for the sequential two-level memory model
  - requires no tuning to cache size
- is **numerically stable**
  - uses partial pivoting (row interchanges)
- uses a matrix data layout that changes on the fly
  - we call it **shape-morphing**

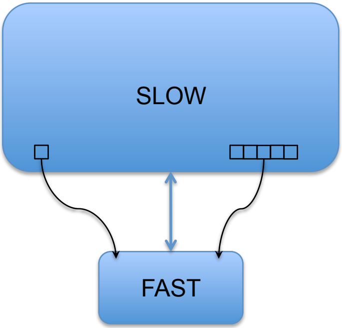
Two-Level Memory Model

- Computation happens only in fast memory (of size M)
- Matrix is too large to fit in fast memory
- Communication happens between slow and fast memory
- Words stored contiguously in slow memory can be read or written as a single message



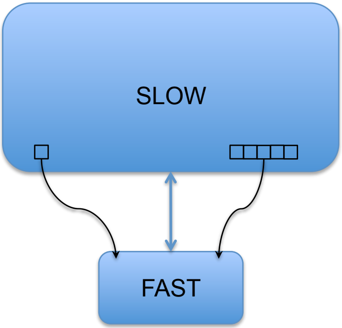
Two-Level Memory Model

- Computation happens only in fast memory (of size M)
- Matrix is too large to fit in fast memory
- Communication happens between slow and fast memory
- Words stored contiguously in slow memory can be read or written as a single message



Two-Level Memory Model

- Computation happens only in fast memory (of size M)
- Matrix is too large to fit in fast memory
- Communication happens between slow and fast memory
- Words stored contiguously in slow memory can be read or written as a single message



$$\text{runtime} = (\#\ \text{messages}) \cdot \alpha + (\#\ \text{words}) \cdot \beta + (\#\ \text{flops}) \cdot \gamma$$

We Have Four Metrics

For best performance on two-level memory model, we want to

(1) minimize words moved
(2) minimize messages moved

We Have Four Metrics

For best performance on two-level memory model, we want to

(1) minimize words moved

(2) minimize messages moved

For best performance on memory hierarchies, we also want to be

(3) cache oblivious

We Have Four Metrics

For best performance on two-level memory model, we want to

(1) minimize words moved

(2) minimize messages moved

For best performance on memory hierarchies, we also want to be

(3) cache oblivious

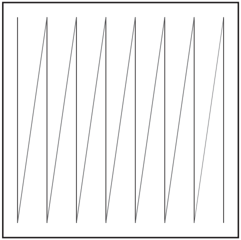
To get the right answer, we need to maintain

(4) numerical stability

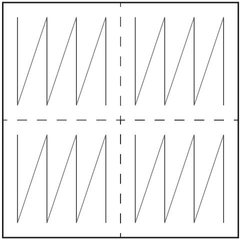
Summary Table

Algorithm	**Minimizes Words**	**Minimizes Messages**	**Cache Oblivious**	**Numerically Stable**
LAPACK [ABB+92]	✗	✗	✗	✓
Square-Recursive LU [BFJ+96]	✓	✓	✓	✗
Comm-Avoiding LU [GDX11]	✓	✓	✗	✓
Rectangular-Rec LU [Tol97]	✓	✗	✓	✓
Shape-Morphing LU	✓	✓	✓	✓

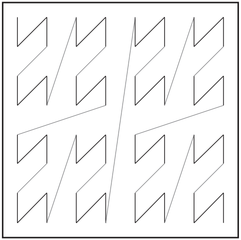
Recall: Matrix Data Layouts



Column Major



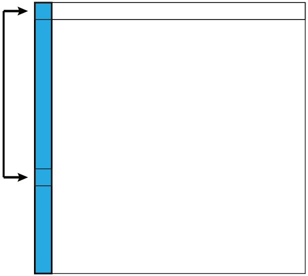
Block Contiguous



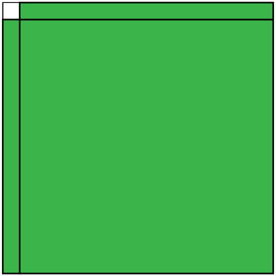
Block Recursive

- column major is most commonly used
- block contiguous has a block size parameter
- block recursive is also known as Morton ordering or bit-interleaved

Recall: (Naive) Gaussian Elimination with Partial Pivoting



Find pivot and scale column

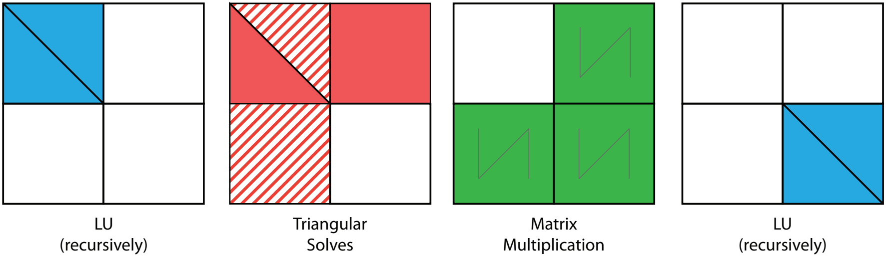


Update Schur complement

for each column:

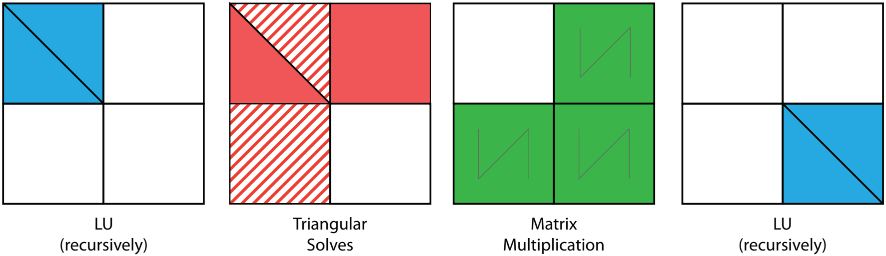
- pivot the largest entry to the diagonal
- divide the column by the diagonal entry
- perform rank-one update on the trailing matrix (Schur complement)

Square-Recursive Algorithm [BFJ+96]



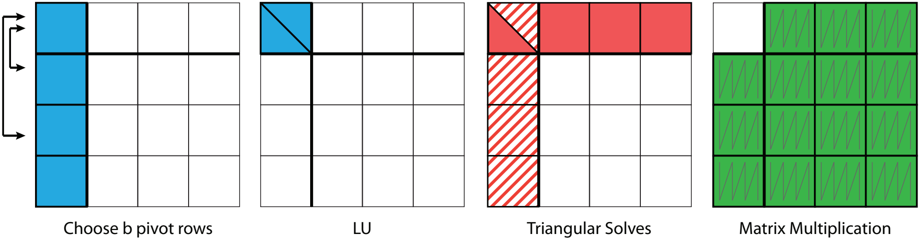
- maps to block recursive layout
- minimizes words and messages and is cache oblivious

Square-Recursive Algorithm [BFJ+96]



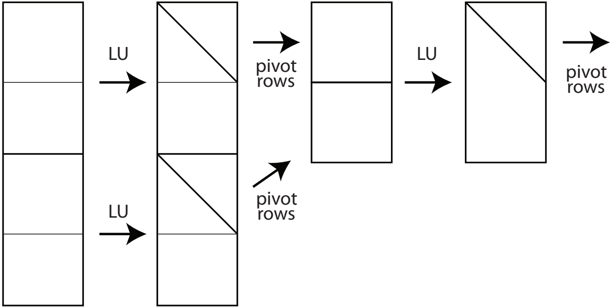
- maps to block recursive layout
- minimizes words and messages and is cache oblivious
- we forgot to pivot!
    - not numerically stable

Communication-Avoiding LU (CALU) Algorithm [GDX11]



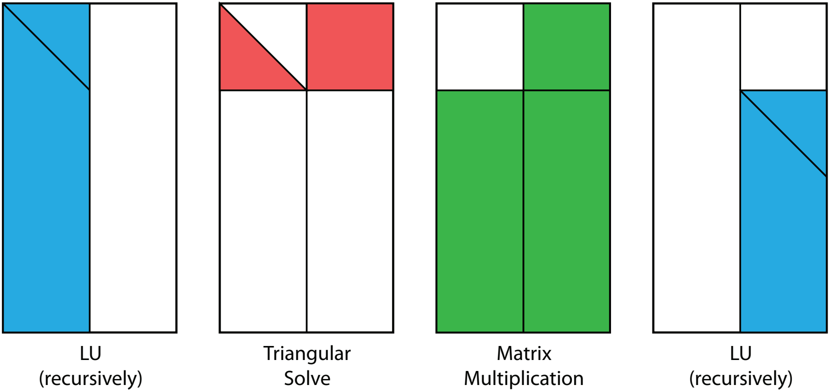
- blocked algorithm, maps to block contiguous layout
- minimizes words and messages, but block size is cache aware
- pivoting scheme is different from partial pivoting, but almost as stable
    - called "tournament pivoting"

Communication-Avoiding LU (CALU) Algorithm [GDX11]



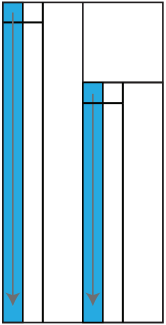
- blocked algorithm, maps to block contiguous layout
- minimizes words and messages, but block size is cache aware
- pivoting scheme is different from partial pivoting, but almost as stable
  - called "tournament pivoting"

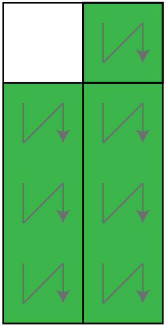
Rectangular-Recursive LU Algorithm [Tol97]



- minimizes words and is cache oblivious
- uses partial pivoting and so is numerically stable
- what data layout to use?

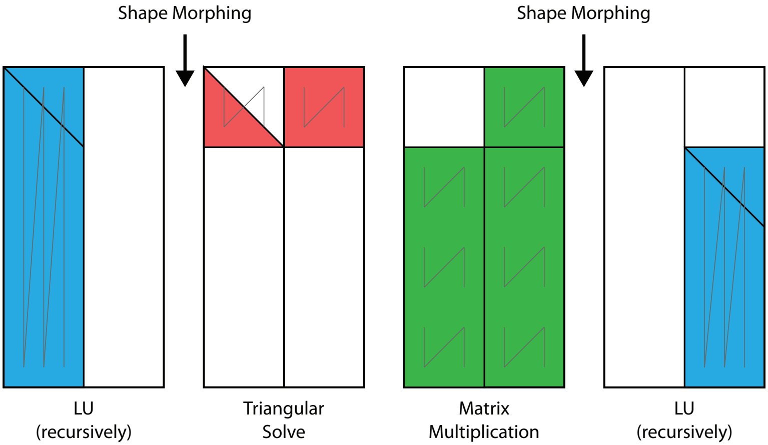
Data Layout Problem





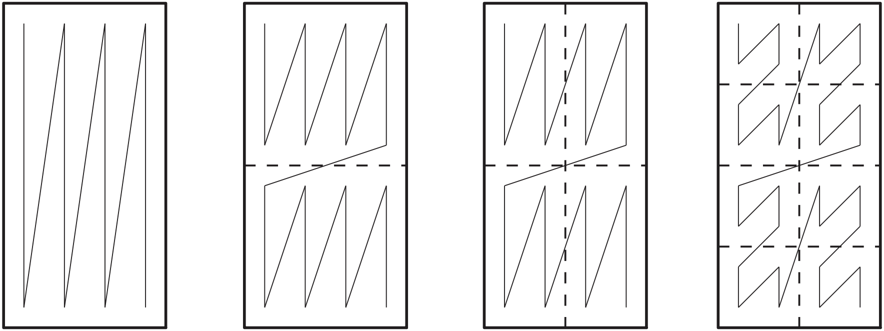
- Base case: find max element in column, pivot, and scale column
  - need column-major layout
  - recursive layout costs too many messages
- Subroutines: rectangular matrix multiplication and triangular solve
  - need recursive layout
  - column-major costs too many messages

Shape-Morphing LU Algorithm



- start and end in column-major layout
- switch to recursive for subroutine calls, then switch back

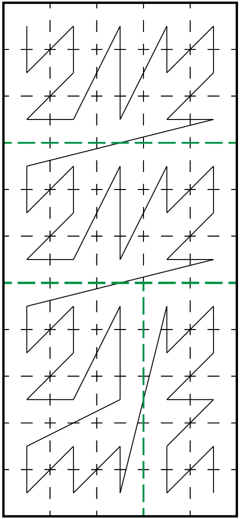
Main Idea: Shape Morphing



- convert between column-major and recursive layouts
- can be cache oblivious and communication efficient
- sacrifice some extra words moved (lower order term) in order to minimize messages

Other Complications

- rectangular recursive layout
  - generalizes Morton ordering
  - "split largest dimension"
- rectangular triangular solve
  - recursive algorithm
- applying pivots (row interchanges)
  - needs to be cache oblivious



Asymptotics

Algorithm	Words	Messages
Lower Bound [BDHS11, GDX11]	$\Omega\left(\frac{n^3}{\sqrt{M}}\right)$	$\Omega\left(\frac{n^3}{M^{3/2}}\right)$
CALU [GDX11]	$O\left(\frac{n^3}{\sqrt{M}} + n^2\right)$	$O\left(\frac{n^3}{M^{3/2}} + \frac{n^2}{M}\right)$
Rect-Rec LU [Tol97]	$O\left(\frac{n^3}{\sqrt{M}} + n^2 \log \frac{n^2}{M}\right)$	$O\left(\frac{n^3}{M} + \frac{n^2}{M} \log \frac{n^2}{M}\right)$
Shape-Morphing LU	$O\left(\frac{n^3}{\sqrt{M}} + n^2 \log^2 \frac{n^2}{M}\right)$	$O\left(\frac{n^3}{M^{3/2}} + \frac{n^2}{M} \log^2 \frac{n^2}{M}\right)$

n is matrix dimension, M is fast memory size

(this table assumes a square matrix, maximum message size of M)

Summary Table

Algorithm	Minimizes Words	Minimizes Messages	Cache Oblivious	Numerically Stable
LAPACK [ABB+92]	✗	✗	✗	✓
Square-Recursive LU [BFJ+96]	✓	✓	✓	✗
Comm-Avoiding LU [GDX11]	✓	✓	✗	✓
Rectangular-Rec LU [Tol97]	✓	✗	✓	✓
Shape-Morphing LU	✓	✓	✓	✓

Discussion / Open Problems

- Same story for QR decomposition
  - shape morphing technique can be applied to a similar rectangular recursive algorithm with equivalent results
- Extension to parallel case is an open problem
  - tournament pivoting seems necessary in parallel case
  - data redistribution on the fly seems too expensive
- Performance data still needed
  - shape morphing will be most useful when latency costs are high and message sizes can be large (e.g., out-of-core computations)

Communication Efficient Gaussian Elimination with Partial Pivoting using a Shape Morphing Data Layout

Grey Ballard, James Demmel, Benjamin Lipshitz, Oded Schwartz, and Sivan Toledo

Thank You!

Funding from Microsoft (Award #024263) and Intel (Award #024894), and matching funding by U.C. Discovery (Award #DIG07-10227). Additional support comes from ParLab affiliates National Instruments, Nokia, NVIDIA, Oracle and Samsung, as well as MathWorks. Research is also supported by DOE grants DE-SC0004938, DE-SC0005136, DE-SC0003959, DE-SC0008700, and AC02-05CH11231, and DARPA grant HR0011-12-2-0016, and grant 1045/09 from the Israel Science Foundation (founded by the Israel Academy of Sciences and Humanities), and grant 2010231 from the US-Israel Bi-National Science Foundation.

References I

E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen.
LAPACK's user's guide.
Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.
Also available from http://www.netlib.org/lapack/.

G. Ballard, J. Demmel, O. Holtz, and O. Schwartz.
Minimizing communication in numerical linear algebra.
SIAM J. Matrix Analysis Applications, 32(3):866–901, 2011.

R. Blumofe, M. Frigo, C. Joerg, C. Leiserson, and K. Randall.
An analysis of dag-consistent distributed shared-memory algorithms.
In *Proceedings of the eighth annual ACM symposium on Parallel algorithms and architectures*, SPAA '96, pages 297–308, New York, NY, USA, 1996. ACM.

L. Grigori, J. Demmel, and H. Xiang.
CALU: A communication optimal LU factorization algorithm.
SIAM Journal on Matrix Analysis and Applications, 32(4):1317–1350, 2011.

S. Toledo.
Locality of reference in LU decomposition with partial pivoting.
SIAM J. Matrix Anal. Appl., 18(4):1065–1081, 1997.